

REMARKS

This application has been carefully reviewed in light of the Office Action dated March 20, 2003. Claims 1-25 remain pending in this application. Claims 2, 9 and 17 have been amended. Favorable reconsideration is respectfully requested.

In the Office Action, Claims 2, 9 and 17 were rejected under 35 U.S.C. 112, second paragraph, as being indefinite. As shown above, these claims have been amended to attend to the points raised in the Office Action. Withdrawal of this rejection is respectfully requested.

Claims 1-3, 6-10, 13-18 and 21-25 were rejected under 35 U.S.C. 103 as being unpatentable over U.S. Patent 6,272,517 (Yue) in view of the article by Sha.

Applicant also notes that Claims 4, 5, 11, 12, 19 and 20 were indicated as allowable if rewritten in independent form. Applicant has refrained from so rewriting these claims because for at least the reasons given below, their respective base claims are believed allowable over the cited art.

Claim 1 is directed to a method for sharing execution capacity among tasks executing in a real-time computing system having a performance specification in accordance with Rate Monotonic Analysis (RMA). The method includes the steps of pairing a higher priority task with a lower priority task, reallocating execution time from the lower priority task to the higher priority task during an overload condition, and increasing the period of the lower priority task to compensate for said reallocated execution time.

In particular, Applicant notes that Claim 1 recites that the reallocation of execution time from the lower priority task to the higher priority task is done during an overload condition. Overload conditions have a significant effect on real-time systems and must be accounted for in scheduling. For example, in a real-time system, it is the worst-case system response time to events that matters. Stability in overload means the system is able to meet its critical deadlines even if all deadlines cannot be met.

One aspect of various embodiments of the present invention is, during an overload condition, to dynamically borrow execution time from the execution capacity of a lower priority task to a higher priority task can without affecting the schedulability of the rest of the system. The higher priority task is bolstered in proportion to the capacity borrowed from the lower priority task, so that the combined utilization of the two tasks remains constant (see page 4, lines 1-7, of the specification). Another aspect of the various embodiments of the present invention is to gracefully degrade system performance during overload while still providing critical schedulability guarantees (see page 5, lines 31-33, of the specification).

Yue, as understood by Applicant, relates to a device for sharing a time quantum when one thread is blocked. The Office Action points out that Yue does not teach reallocation of execution time in response to an overload condition. The Office Action cites Sha as disclosing transforming the execution period of a task in response to an overload condition (i.e., at pages 258-59).

However, Applicant respectfully submits that the portion of Sha cited to merely reinforces the need for the present invention. For example the definition of a stable

scheduling algorithm specifically cited in the Office Action is one reason the why the present invention is needed. Sha merely discussed the period transformation technique that turns a long-period important task into a high priority task by splitting its work over several short periods. Nothing found in Sha teaches or suggests reallocating execution time from the lower priority task to the higher priority task during an overload condition.

In addition, even if it were deemed obvious to one of ordinary skill in the art to increase the period of a lower priority task, as the Office Action suggests, this does not teach or suggests that this should be done during an overload condition. In this regard, Sha teaches that the period transformation scheduling technique should be done to prevent possible overload conditions – it does not teach how to overcome or address overload conditions that actually happen.

In view of this, Applicant respectfully submits that one of ordinary skill in the art would not combine Yue and Sha as suggested in the Office Action because (1) Yue relates to sharing a time quantum during blocking – not overload and (2) Sha does not teach dynamic allocation of execution time during an overload condition.

At least for these reasons, Claim 1 is believed patentable over Yue and Sha.

The other rejected independent claims recite a feature similar as discussed above in regard to Claim 1 and are believed patentable for at least the same reasons.

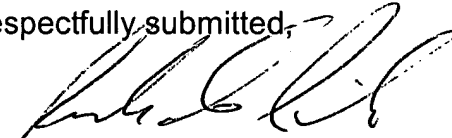
The other rejected claims in this application are each dependent from one or the other of the independent claims discussed above and are therefore believed patentable for the same reasons. Since each dependent claim is also deemed to define an additional

aspect of the invention, however, the individual reconsideration of the patentability of each on its own merits is respectfully requested.

In view of the foregoing submission and remarks, Applicant respectfully requests favorable reconsideration and early passage to issue of the present application.

Applicant's attorney may be reached by telephone at the number given below.

Respectfully submitted,

By 
Rick de Pinho, Reg. No. 41,703

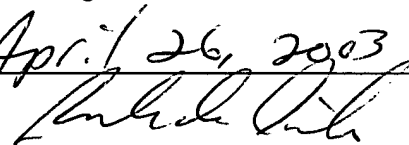
Mail all correspondence to:
US PHILIPS CORPORATION
580 White Plains Road
Tarrytown, NY 10591
Tel: (914) 333-9609

For Tony E. Piotrowski, Reg. No 42,080
Attorney for Applicants

CERTIFICATE OF MAILING

It is hereby certified that this correspondence is being deposited with the United States Postal Service as first-class mail in an envelope addressed to:

COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

On Apr. 26, 2003
By 
Rick de Pinho, Reg. 41,703

Appendix of Marked-up Claim Amendments

2. (Amended) The method of claim 1, wherein an amount of said execution time available to loan from said lower priority task_r (hereinafter "task_r") to said higher priority task_u (hereinafter "task_u") is obtained as follows:

$$N_u = \frac{N_r \cdot T_u}{T_r}$$

where,

N_r = amount of execution time to borrow from task_r, where $N_r < C_r$,

T_r = period of task_r,

C_r = worst-case task execution time of task_r and

T_u = period of task_u.

9. (Amended) The method of claim 8, wherein said reallocated portion of said first resource allocation is obtained as follows:

$$N_u = \frac{N_r \cdot T_u}{T_r}$$

where,

N_r = amount of execution time to borrow from task_r, where $N_r < C_r$,

T_r = period of the lower priority task ("task_r"),

C_r = worst-case task execution time of task_r and

T_u = period of the higher priority task ("task_u").

17. (Amended) The method of claim 15, wherein an amount of said execution time available to reallocate from said lower priority task_r (hereinafter "task_r") to said higher priority task_u (hereinafter "task_u") is obtained as follows:

$$Nu = \frac{Nr \cdot Tu}{Tr}$$

where,

N_r = amount of execution time to borrow from task_r, where $N_r < C_r$,

T_r = period of task_r,

C_r = worst-case task execution time of task_r, and

T_{uU} = period of task_{uU}.